ELSEVIER

Theoretical Computer Science 218 (1999) 263–271

Theoretical Computer Science

Non-uniform random spanning trees on weighted graphs[1]

M. Mosbah, N. Saheb *

LaBRI[2], Université Bordeaux-I, 351, cours de la Libération, 33405 Talence, France

Abstract

We study random walks on undirected graphs with weighted edges. Our main result shows that any spanning tree defined by the edges corresponding to a first visit of a vertex, appears with a probability proportional to its weight, which is the product of the weight of its edges. This provides an algorithm for generating non uniform random spanning trees in a weighted graph. The technique used here is based on linear equations over regular expressions and finite automata theory. © 1999 Elsevier Science B.V. All rights reserved.

Résumé

Nous étudions les marches aléatoires sur les graphes pondérés non orientés. Nous démontrons, en particulier, que la probabilité d'engendrer un arbre couvrant est proportionnelle à son poids, qui est le produit de poids de ses arêtes. Ce résultat fournit un algorithme pour générer des arbres couvrants non uniformes dans un graphe pondéré. Les techniques utilisées sont fondées sur les systèmes d'équations sur les langages rationnels et la théorie des automates finis. © 1999 Elsevier Science B.V. All rights reserved.

1. Introduction

Random walks have been studied extensively, and have many applications such as generation of random spanning trees [1, 2, 12], token management schemes [8, 11], effective resistance of electrical networks [3, 5, 10], and on-line algorithms [4].

In this paper, we consider a connected simple undirected graph $G=(V,E)$ together with a positive real-valued map w over E, $w(e)$ being called the weight of e. A discrete time random walk (or Markov chain) on G is defined as follows. At each step,

* Corresponding author. E-mail: saheb@labri.u-bordeaux.fr.

[1] This work has been supported by the ESPRIT Basic Research Working Group "Computing by Graph Transformations II".

[2] Laboratoire associé au CNRS.

PII: S0304-3975(98)00325-9

a particle, located on a vertex x, moves to a neighbour vertex y with a probability proportional to the weight of the edge (x, y), that is, the probability of the transition from x to a neighbour y is $w((x,y))/\sum_{z\in N(x)} w((x,z))$, where $N(x)$ is the set of neighbours of x. The generated spanning tree by the random walk is the spanning tree consisting of those edges which correspond to the first entrance to an arbitrary vertex of G, other than the starting vertex. The main goal of this paper is to introduce syntactic tools to study specific problems in random walks. More applications of these techniques, such as mean cover time and mean hitting time computation are available in [9], in which we provide methods based on syntactic approach to compute the mean cover time, i.e. the expected number of necessary transitions for the particle to visit all vertices, and the mean hitting time, i.e. the expected necessary transitions for reaching a given vertex from another one. Another interesting feature of this approach, discussed in [9], is that it allows to distinguish explicitly easy problems from difficult ones according to their syntactic complexities.

In the case of uniform random walks (where $w(e)=1$, for any $e\in E$), Aldous and Broder proved that all spanning trees of G have the same probability of being generated, whatever is the starting vertex.

For non uniform random walks, the set of walk behaviours providing a given spanning tree, for a fixed starting vertex, can be modeled by a regular expression, characterized by a linear system of equations. Translating this system into an arithmetic system of equations over the corresponding probabilities, we prove that the probability of generating a tree is proportional to the product of the weights of its edges. We also find the stationary probability of a vertex, i.e. the probability that a random walk is currently at such a vertex. This probability is proportional to the weight of this vertex (i.e. the sum of the weights of the incident edges to the vertex).

We consider the case of a weighted cycle. A random walk can be viewed as a particle that goes randomly from a vertex to one of its two neighbours. Once the particle has visited all vertices, it has gone across all edges of the cycle except one, which has been left out. We prove that the probability of an edge to be left out does not depend on the starting vertex. Its probability is proportional to the inverse of its weight.

The paper is organized as follows. In Section 2, we introduce some notations and definitions related to words and random walks. The probability of a generated spanning tree is discussed in Section 3. In this section, we also investigate the stationary probability of a vertex. Section 4 deals with the class of cycle graphs. In particular we compute the probability for an edge of a cycle to be left out during a random walk. Section 5 presents some open problems and further extensions of this work.

2. Notations and definitions

We denote by $G=(V,E)$ a simple undirected connected weighted graph. Weights are positive reals.

A *finite walk* over G is a finite sequence $w = i_1, i_2, \ldots, i_m$ of vertices of G such that $\{i_k, i_{k+1}\} \in E$, $i = 1, \ldots, m-1$. An *infinite walk* over G is an infinite sequence $i_1, i_2, \ldots$, with $\{i_k, i_{k+1}\} \in E$. A walk may also be viewed as a sequence of directed edges $(i_1, i_2), (i_2, i_3), (i_3, i_4) \ldots$. Thus, if we introduce an alphabet A whose letters are elements of E on which we consider two possible directions, a finite (infinite) walk, according to this view, is a word (infinite word) over A. The set of infinite walks on G starting with vertex i will be denoted by W_i. The *generated tree* $\tau(w)$ by the walk w is the set of all edges corresponding to the *first* entrance of the walk into a vertex of G [1, 2]; it is easy to see that $\tau(w)$ is an undirected tree. For a given vertex i and a tree T containing i, the language $L_i(T)$ denotes the set of shortest finite walks (in the prefix order) starting with i and generating T, i.e.

$$L_i(T) = \{w \mid w \text{ is a finite walk starting with i, such that } \tau(w) = T \text{ and for no proper left factor } w' \text{ of } w,\ \tau(w') = T\}.$$

We sketch in the next subsection the construction of a finite automaton $\mathcal{A}_i(T)$, which recognizes $L_i(T)$. Thus $L_i(T)$ is a regular language. The elements of $L_i(T)$ can be extended into infinite walks in a natural way, more precisely

$$\overline{L_i}(T) = \{ww' \mid w \in L_i(T),\ w' \in W_j \text{ where } j \text{ is the last vertex of } w\}.$$

Intuitively, $\overline{L_i}(T)$ consists of infinite walks starting with i and generating T.

2.1. A recognizing automaton

Consider in $G = (V, E)$ a given vertex $i \in V$ taken as the starting vertex and a spanning tree T. For a nonempty subset X of V, let $G_X = (X, E_X)$ denote the subgraph of G induced by X, i.e. $E_X = \{(x, y) \mid x \in X, y \in X\}$. The finite deterministic automaton $\mathcal{A}_i(T)$, recognizing $\overline{L_i}(T)$, is introduced as follows. The set Q of states is the set of all ordered pairs (X, j), with $j \in X$, where X is any set of vertices containing i such that the induced subgraph G_X is connected. Roughly speaking, in a walk over G, the state (X, j) corresponds to the fact that j is the currently visited vertex and that X is the set of already visited vertices. The initial state of $\mathcal{A}_i(T)$ will be $(\{i\}, i)$. The terminal states of $\mathcal{A}_i$ are those of the form (V, j), $\forall j \in V$. These states correspond to the event that all vertices of G have been visited, and thus as we shall see, no transitions are defined for them. We now define the transition (partial) function θ over Q as follows.

1. For any pair of states $r = (X, j)$ and $s = (X, k)$, with $X \neq V$, such that j and k are neighbours in G, we let $\theta(r, a) = s$ (resp. $\theta(s, b) = r$), where $a \in A$ (resp. $b \in A$) identifies the directed edge $(j, k) \in E_x$ (resp. (k, j)).
2. For a pair of states $r = (X, j)$ and $s = (X \cup \{k\}, k)$ with $k \notin X$, such that $\{j, k\} \in T$, we let $\theta(r, a) = s$, where $a \in A$ identifies the the directed edge (j, k).

Let us notice that it is possible to reduce the set of terminal states into a unique finite state

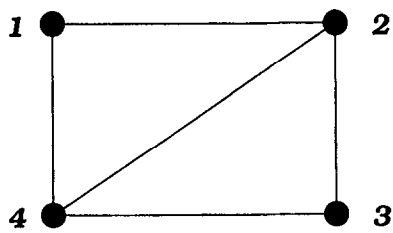

Fig. 1. A graph.

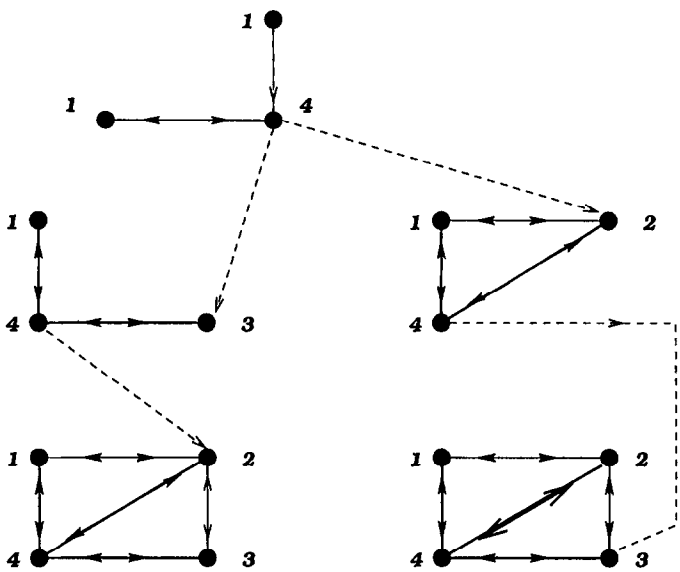

Fig. 2. An automaton recognizing $\overline{L_1}(T)$.

Example 1. Consider the graph of Fig. 1. Let T be the tree $\{\{4,1\},\{4,2\},\{4,3\}\}$. For the starting vertex 1, the automaton $\mathcal{A}_1(T)$ is given by Fig. 2. Dashed transitions correspond to first entrances. Edges containing arrows in both directions can be scanned by the walk following these directions. Bold edges belong to the tree T. Vertex 2 on the left side is the last visited vertex before generating T. Similarly for vertex 3 on the right side. $L_1(T)$ is the set of all finite paths beginning at the starting arrow and ending with one of the exit arrows. $\overline{L_1}(T)$ is the set of all infinite paths beginning with the starting arrow and going infinitely many times through one of the vertices of the bottom rectangles labeled with 2. It is clear that the language of infinite words $\overline{L_i}(T)$ is recognizable [6, Chap. XIV]. $L_i(T)$ and its recognizing automaton can be used to compute the cover time (i.e. the expected necessary time to visit all vertices of G), see [9].

We now introduce useful definitions related to probability measures on the walks. For a vertex i, let $w(i)$, called the *weight* of i, be the sum of the weights of the incident edges to i. The weight of a tree T is defined by the product of the weights of its edges. The probability measure associated with the directed edge (i,j) is $p_{ij} = w(i,j)/w(i)$; it is the probability of a transition from i to j, and we have obviously $\sum_{j\in N(i)} p_{ij} = 1$, where $N(i)$ is the set of neighbour vertices of i. The probability of a finite walk $w = (i_0,i_1)(i_1,i_2),\ldots,(i_{m-1},i_m)$ is the product of the probability of its directed edges and it will be denoted by $p(w)$.

We have thus a finite state Markov chain whose set of states is V and whose transition (stochastic) matrix is given by $p_{ij} = w(i,j)/w(i)$, $i,j \in V$. For a given walk $w = (i_0,i_1)(i_1,i_2),\ldots,(i_{m-1},i_m)$, its probability $p(w)$ is the conditional probability $Pr(X_1 = i_1,\ldots,X_n = i_n \mid X_0 = i_0)$.

Finally, for a vertex i and a tree T, we define $p_i(T) = \sum_{w\in L_i(T)} p(w) = p(L_i(T))$. It is easy to see that, for a spanning tree T, $p_i(T)$ is the probability of generating T, whenever the walk starts at i. For so, it suffices to see that words of $L_i(T)$ correspond to pairwise disjoint events which generate T, and that T cannot be generated by anything else than these events. Since the elements of $\overline{L}$ are obtained from those of L by adding all possible extensions, $\overline{L}$ and L are of the same probability and therefore $p_i(T) = p(\overline{L_i}(T))$. This probability can be interpreted in the Markov chain as follows. Any orbit of the random process generates a tree T in G, which is the set of edges corresponding to a the first entrance into some state. According to this interpretation, $p_i(T)$ is the probability of generating T, conditioned by the event that the initial state is i.

3. Random spanning trees

In the sequel, the set of trees which are subgraphs of G is denoted by $\mathscr{T}$ and the set of spanning trees of G by $\mathscr{S}$.

Lemma 3.1. *For any vertex i, we have*

$$\sum_{T\in\mathscr{S}} p_i(T) = 1.$$

Proof. Starting from i, all vertices of G will finally be visited with probability 1, because there is only one class of states (see for instance [7, Chap. XV]). On the other hand, the languages $L_i(T)$, $T \in \mathscr{S}$, are pairwise disjoint sets. □

Let i be a vertex and T a spanning tree in G. For a neighbour vertex of j such that $\{i,j\} \notin T$, the set $T \cup \{\{i,j\}\}$ contains a unique cycle. In the following lemma, we denote by $\overline{j}$ the unique vertex such that $\{i,\overline{j}\}$ belongs to T and the cycle, and $T_j = T \cup \{\{i,j\}\}\backslash\{\{i,\overline{j}\}\}$.

Lemma 3.2. *For i and T as above, we have*

$$\overline{L_i}(T) = \sum_{j:(i,j)\in T} (i,j)\overline{L_j}(T) + \sum_{j\in N(i):(i,j)\notin T} (i,\bar{j})\overline{L_{\bar{j}}}(T_j) + K_i(T),$$

where $K_i(T) = \emptyset$ if the degree of i is greater than 1 in T, otherwise $K_i(T) = (i,j)\overline{L_j}(T')$ with $T' = T\backslash\{\{i,j\}\}$, where j is the unique adjacent vertex to i in T.

Proof. It is clear that the left member of the equation is a subset of the right one. A simple verification case-by-case allows to show that the right member is also a subset of the first one. □

Example 2. Consider again the graph given in Example 1.

Let $T = \{\{1,2\},\{2,4\},\{4,3\}\}$, then

$$\overline{L_2}(T) = (2,1)\overline{L_1}(T) + (2,4)\overline{L_4}(T) + (2,4)\overline{L_4}(\{\{1,2\},\{2,3\},\{3,4\}\}),$$

since clearly:

$$K_2(T) = \emptyset.$$

For the initial vertex 1 and the same tree T, we have

$$\begin{aligned}\overline{L_1}(T) = {} & (1,2)\overline{L_2}(T) + (1,2)\overline{L_2}(\{\{2,4\},\{3,4\},\{1,4\}\}) \\ & + (1,2)\overline{L_2}(\{\{2,4\},\{3,4\}\}).\end{aligned}$$

We now state the main result.

Proposition 3.1. *The probability of generating a given spanning tree is proportional to its weight.*

Proof. The previous lemma provides $|V| \times |\mathscr{S}|$ equations over elements of A^∞ (A^∞ is the set of infinite words over A). Since the terms of the sum are pairwise disjoint, the equations on the languages can be transformed into equations on probabilities. Using the fact that for a non spanning tree U, the probability $p(\overline{L_i}(U)) = 0$, we get $\forall i \in V,\ \forall T \in \mathscr{S}$

$$p_i(T) = \sum_{j:(i,j)\in T} \frac{w(i,j)}{w(i)} p_j(T) + \sum_{j\in N(i),(i,j)\notin T} \frac{w(i,\bar{j})}{w(i)} p_{\bar{j}}(T_j).$$

Now, by Lemma 3.1, we must have also

$$\forall i \in V,\ \sum_{T\in\mathscr{S}} p_i(T) = 1.$$

Clearly, the vector $p_i(T) = w(T)/\sum_{T'\in\mathscr{S}} w(T')$ is a solution of the above system of linear equations. It remains, therefore, to prove that the solution is unique. Let us consider the finite Markov chain with the set of states $\{(i,T) \mid i \in V, T \in \mathscr{S}\}$ whose

stationary probabilities are a solution of the above system of linear equations. For showing the uniqueness of the solution, it suffices to prove that the Markov chain is irreducible [7, Chap. XV]. The first terms of the first equations show that, for a fixed T, a k-step transition between two states (i,T) and (i',T) is possible for any pair of vertices i and i'. The second terms of the first equations correspond to a transition of the type $(i,T) \leftarrow (\bar{j},T_j)$ and show that a transition $(.,T_1) \leftarrow (.,T_2)$ is possible whenever T_1 and T_2 differ only by one edge. Since any spanning tree may be obtained from any other one by a sequence of insertion-deletion, the introduced chain will be irreducible and the theorem follows. □

During a random walk, a vertex can be visited many times. The question that we shall answer now, is what is the probability that the current visited vertex is actually a given vertex i? We shall show that this probability is proportional to the weight of i. Let π be the stationary distribution over the set of vertices. That is, $\pi(i)$ is the probability that a random walk, that has begun at any other vertex, is currently at vertex i.

Proposition 3.2. *The stationary probability of any vertex i is proportional to its weight. Indeed,*

$$\pi(i) = \frac{w(i)}{\sum_{j \in V} w(j)}.$$

Proof. The vector π, satisfies the system

$$\pi(i) = \sum_{j \in N(i)} p_{ji}\pi(j), \quad \forall i \in V.$$

We also have the equation $\sum_{i \in V} \pi(i) = 1$. Using similar techniques as above, we can show that the vector $\pi(i) = w(i)/\sum_{j \in V} w(j)$, $i \in V$, is the unique solution of the system. This ends the proof. □

Note that, for the particular case where the weight of any edge is 1, the stationary probability of vertex i is $d(i)/2|E|$ where $d(i)$ is the degree of i and $|E|$ is the cardinality of the set of edges.

4. Cycle graph

Let G be an undirected connected cycle, called also a ring. Consider a particle that moves on G. At each step, the particle goes from the current vertex to one of its two neighbours, with a probability proportional to the weights of the two incident edg. Starting at a vertex, it is clear that once the particle has visited all vertices, it has gone through all edges in the cycle except one, called the *left-out edge* [2]. Surprisingly, even for weighted graphs, the probability of an edge to be the left-out one, does not depend on the starting vertex; indeed it is proportional to the inverse of its weight.

Proposition 4.1. *Let $G=(V,E)$ be a cycle graph, and let e be an edge in G. Then, the probability $\overline{p}(e)$ for e to be left-out is*

$$\overline{p}(e)=\frac{1}{\sum_{f\in E}\frac{1}{w(f)}}\frac{1}{w(e)}.$$

Proof. This is a consequence of the main proposition. In fact, the probability of an edge e to be left-out is the same as that of the spanning tree containing all other edges. Writing the probability of generating such a tree, we obtain the result. □

Note that for the particular case where the weight edge is 1, the left-out edge induced by a random walk is uniformly distributed over the edges of the cycle.

5. Conclusion

In this paper we used regular expressions and automata to deal with random walks on weighted graphs. The advantage of using automata is to memorize already visited vertices. A transition is created as soon as a new vertex is visited. Therefore, the execution of the automaton generates a random spanning tree. An interesting extension of this work is to use this technique to compute the cover time of a random walk on a weighted graph, i.e. the expected time to visit all vertices at least once, which is also the expected time to generate a random spanning tree. One other useful extension would be to investigate cover time of random walks on particular classes of weighted graphs. For example, trees, cycles, chains and cliques are classes of graphs for which the cover time seems to be efficiently computable, (see [9] for more details).

Acknowledgements

The authors would like to express their gratitude to Bertrand Le Saëc for helpful discussions.

References

[1] D.J. Aldous, The random walk construction of uniform spanning trees and uniform labelled trees, SIAM J. Discrete Math. 3 (4) (1990) 450–465.

[2] A.Z. Broder, Generating random spanning trees, in: Proc. 30th Ann. IEEE Symp. on Foundations of Computer Science, October 1989, pp. 442–453.

[3] A.K. Chandra, P. Raghavan, W.L. Ruzzo, R. Smolensky, P. Tiwari, The electrical resistance of a graph captures its commute and cover times, in: ACM Symp. on Theory of Computing (STOC), 1989, pp. 574–586.

[4] D. Coppersmith, P. Doyle, P. Raghavan, M. Snir, Random walks on weighted graphs and applications to on-line algorithms, J. ACM 40 (3) (1993) 421–453.

[5] P.G. Doyle, J.L. Snell, Random walks and electrical networks, The Mathematical Association of America, 1984.

[6] S. Eilenberg, Automata, Languages, and Machines, vol. A, Academic Press, New york, 1974.
[7] W. Feller, An introduction to probability theory and its applications, vol. 1, 2nd ed., Wiley, New York, 1957.
[8] A. Israeli, M. Jalfon, Token management schemes and random walks yield self-stabilizing mutual exclusion, in: Proc. 9th Annual Symp. on Principles of Distributed Computing, 1990, pp. 119–131.
[9] M. Mosbah, N. Saheb, A syntactic approach to random walks on graphs, Technical Report 1163-97, University of Bordeaux 1, 1997, WG'97, to appear.
[10] P. Tetali, Random walks and the effective resistance of networks, J. Theoret. Probab. 4 (1991) 101–109.
[11] P. Tetali, P. Winkler, On a random walk arising in self-stabilizing token management, in: Proc. 10th Annual ACM Symp. on Principles of Distributed Computing, 1991, pp. 273–280.
[12] D.B. Wilson, J.G. Propp, How to get an exact sample from a generic Markov chain and sample a random spanning tree from a directed graph, both within the cover time?, in: Proc. 7th Annual ACM-SIAM Symp. on Discrete Algorithms, Atlanta, GA, 28–30 January 1996, pp. 448–457.